

Cursos: Sistemas de Informação

Disciplina : Programação Orientada a Objetos – 2021/02

Professora : Nádia Félix

Data : 06/09/2021

Aluno(a) : _____

1. Responda as questões abaixo **(2,0)**:

- a) Conceito da orientação a objetos que está relacionado a esconder as características e publicar os comportamentos de um objeto.
- b) A possibilidade de métodos com a mesma assinatura terem comportamentos diferentes em uma hierarquia de classes está relacionada a que conceito da POO?
- c) A possibilidade de reutilização de código permitindo a criação de uma classe aproveitando atributos e métodos de outra está relacionada a que conceito da POO?
- d) Como são chamados os atributos e métodos que são acessados sem a necessidade de criação de um objeto?
- e) Podem existir dois métodos, na mesma classe, com o mesmo nome e comportamentos diferentes? Em que situação?

2. Aplicando as boas práticas no código abaixo, ou seja, escondendo os atributos e publicando métodos de acesso, quais alterações devem ser feitas nas classes **Revista** e **Principal** para que o programa continue funcionando da mesma forma? **(2,0)**:

Obs1: Só o necessário

Obs2: Considere que exista uma classe Data com dia, mês e ano e com um construtor conforme invocado no código.

```
public class Revista {
    String nome;
    Data dataPublicacao;
    String editora;

    Revista (String nome, String editora) {
        this.nome = nome;
        this.editora = editora;
    }
}

public class Principal {

    public static void main(String[] args) {

        Revista revista = new Revista("Casa Claudia", "Abril");

        revista.dataPublicacao = new Data(10,01,2015);

        System.out.println ("Revista : " + revista.nome + " " + revista.editora);

    }
}
```

Cursos: Sistemas de Informação

Disciplina : Programação Orientada a Objetos – 2021/02

Professora : Nádia Félix

Data : 06/09/2021

Aluno(a) : _____

3. Faça o que se pede em cada item **(2,0)**:

- criar uma classe Estado com os atributos **privados**, nome (String) e uf (String).
- criar um **construtor** que receba como parâmetros um nome e uma uf e atualize os atributos do objeto.
- criar os **métodos de acesso GET** para nome e uf
- criar um programa principal com o método main para instanciar 2 objetos do tipo Estado:
 - UF: GO Nome: Goiás
 - UF: RJ Nome: Rio de Janeiro
- Implemente, no método main, uma forma de mostrar os dois estados criados usando os **métodos de acesso**.

4. Considere a classe Apartamento abaixo, com seus atributos e métodos, para atualização dos dados e para mostrar o apartamento **(4,0)**:

Obs: Não precisa usar classes de entrada de dados (Scanner, JOptionPane), pode considerar dados fixos.

```
class Apartamento {
    private int qtdeQuartos;
    private int qtdeBanheiros;
    private String setor;

    Apartamento (int quartos, int banheiros, String setor) {
        qtdeQuartos = quartos;
        qtdeBanheiros = banheiros;
        this.setor = setor;
    }

    public String mostraApartamento () {
        return "Apartamento de " + qtdeQuartos + " quartos, " + qtdeBanheiros + " banheiros, " + " no setor " + this.setor;
    }
}
```

- criar uma classe *Aparthotel*, herdando da classe *Apartamento* e tendo também como atributo, a *rentabilidadeMensal* (valor do possível aluguel mensal).
- criar um construtor para a classe *Aparthotel* que chame o construtor de *Apartamento*.
- criar outro método *mostraApartamento* sobrescrito, na classe *Aparthotel* chamando este mesmo método da superclasse e acrescentando uma linha, no retorno do método, com a seguinte informação "Apart Hotel / Flat".
- criar uma classe *Principal* que terá um único vetor para armazenar tanto objetos do tipo Apartamento como Aparthotel
- instanciar 2 objetos do tipo *Apartamento* usando dados fictícios e armazená-los no vetor
- instanciar 2 objetos do tipo *Aparthotel* usando dados fictícios e armazená-los no vetor
- varrer o vetor para mostrar os dados dos objetos armazenados.
- criar um atributo estático na classe Apartamento para guardar a quantidade de objetos criados e fazer a alteração necessária no construtor para que a cada objetos criado essa quantidade seja incrementada